Discrete Optimization 8 (2011) 129–138

Contents lists available at ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

On the algorithmic effectiveness of digraph decompositions and complexity measures☆

Michael Lampis [a,*], Georgia Kaouri [b], Valia Mitsou [a]

[a] Department of Computer Science, Graduate Center, City University of New York, 365 5th Avenue, New York, NY 10016, USA

[b] Computation and Reasoning lab, School of Electrical and Computer Engineering, National Technical University of Athens, Greece

ARTICLE INFO

Article history:
Received 30 November 2009
Received in revised form 12 March 2010
Accepted 15 March 2010
Available online 1 May 2010

Keywords:
Treewidth
Digraph decompositions
Parameterized complexity

ABSTRACT

We place our focus on the gap between treewidth's success in producing fixed-parameter polynomial algorithms for hard graph problems, and specifically Hamiltonian Circuit and Max Cut, and the failure of its directed variants (directed treewidth (Johnson et al., 2001 [13]), DAG-width (Obdrzálek, 2006 [14]) and Kelly-width (Hunter and Kreutzer, 2007 [15]) to replicate it in the realm of digraphs. We answer the question of why this gap exists by giving two hardness results: we show that Directed Hamiltonian Circuit is $W[2]$-hard when the parameter is the width of the input graph, for any of these widths, and that Max Di Cut remains NP-hard even when restricted to DAGs, which have the minimum possible width under all these definitions. Along the way, we extend our reduction for Directed Hamiltonian Circuit to show that the related Minimum Leaf Outbranching problem is also $W[2]$-hard when naturally parameterized by the number of leaves of the solution, even if the input graph has constant width. All our results also apply to directed pathwidth and cycle rank.

1. Introduction

Treewidth, first introduced by Robertson and Seymour in [1], has been one of the most successful tools in the research for efficient algorithms for hard graph problems for the last 15 years. Intuitively, treewidth allows us to distinguish graphs that have a relatively simple (tree-like) structure, and exploit that structure to solve a plethora of otherwise intractable problems, usually by employing a dynamic programming technique. In addition, treewidth has proven very interesting from a graph-theoretic perspective, one of its most important attributes being that it can be equivalently defined in many seemingly unrelated ways. For example treewidth is connected to chordal graphs, elimination schemes, partial k-trees, cops-and-robber games [2,3], reduction rules [4] and brambles [2]. Thus, treewidth has proven so algorithmically successful and graph-theoretically robust that it is widely considered the "right" complexity measure for undirected graphs. For an introduction to the notion of treewidth see Bodlaender's excellent survey papers [5–7]. For more recent results on the important role treewidth plays in a variety of fundamental combinatorial problems see [8].

One of the most celebrated theorems in the area of treewidth is Courcelle's theorem which states that every graph property that can be expressed in monadic second order logic can be decided in linear time on graphs of bounded treewidth [9]. Beginning from this starting point, algorithms for many hard graph problems have been devised using treewidth. They almost invariably have running times of the form $O(f(k) \cdot n)$, where k is the treewidth of the input graph and f is some exponential or super-exponential function which represents the complexity of solving the problem exhaustively on

☆ A preliminary version of this paper was presented at ISAAC 2008.

* Corresponding author.
E-mail addresses: mlampis@gc.cuny.edu (M. Lampis), gkaouri@corelab.ntua.gr (G. Kaouri), vmitsou@cs.gc.cuny.edu (V. Mitsou).

1572-5286/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2010.03.010

k vertices. Thus, not only is the running time polynomial for fixed k, but also the combinatorial explosion is confined to k. This has led treewidth to become one of the cornerstones of parameterized complexity theory, a theory which describes the distinction between algorithms with running times of the form $O(f(k) \cdot n^c)$, where c is a constant (called fixed-parameter tractable or FPT) and algorithms of the form $O(n^{g(k)})$. For an introduction to parameterized complexity see the monograph by Downey and Fellows [10] or the introductory books by Niedermeier [11] and by Flum and Grohe [12].

Several attempts have been made recently to generalize the notion of treewidth to directed graphs. The motivation behind this line of research is that, although it is possible to solve many hard problems on digraphs when the underlying undirected graph has low treewidth by using traditional tree decompositions, this approach sacrifices a great deal of generality. A problem which demonstrates this to a great degree is Directed Hamiltonian Circuit. This problem is trivial when the input graph is a DAG, but there exist DAGs of unbounded treewidth if the direction of the edges is ignored. Thus, it is desirable to come up with an alternative measure of digraph complexity which better characterizes the class of digraphs where hard problems become tractable. It should be noted at this point that Hamiltonian Circuit admits an FPT solution with a treewidth-based algorithm, therefore a logical target when defining a digraph complexity measure would be to achieve fixed-parameter tractability for Directed Hamiltonian Circuit as well.

1.1. Previous work

The most notable variations of treewidth for digraphs that have been proposed in the past are probably directed treewidth [13], DAG-width [14] and Kelly-width [15]. All these three measures can be viewed as good generalizations of treewidth in the sense that, if we take an undirected graph and replace each edge with two opposite directed edges the width of the new digraph will be the same for all three definitions and equal to the treewidth of the original graph. Directed treewidth is the most general of the three, in the sense that a graph of bounded Kelly-width or DAG-width will also have bounded directed treewidth, while the converse may not be true. Also DAG-width and Kelly-width are conjectured to be only a constant factor apart on any graph [15].

Thus, Kelly-width and DAG-width have the potential to provide better algorithmic properties than directed treewidth and some evidence is given in this direction in the form of an algorithm for solving a class of parity games, a problem that is open so far for directed treewidth (note though that this algorithm is not FPT, and that the problem is not believed to be NP-complete).

The most important positive result of directed treewidth (which can be extended to all the three measures) is an algorithm that solves Directed Hamiltonian Circuit in $O(n^{f(k)})$ time, k being the width of the input graph. Nevertheless, this algorithm is still far from the performance of the best treewidth-based algorithm for Hamiltonian Circuit, which runs in fixed-parameter linear time. Unfortunately, the reason for this distance is not addressed in [13] or in [15] where another algorithm (of similar complexity) for this problem is given. In addition, the few already known algorithmic results on these measures do not seem to indicate that they are likely to achieve a level of success comparable to treewidth, as no FPT algorithms are known for any hard digraph problems. Of course, it could be conceivable that this is due to a lack of effort so far, since digraph decompositions have been introduced much more recently than treewidth.

More recently in [16] Kreutzer and Ordyniak investigate the concepts of DAG-width and Kelly-width more closely and prove several interesting results: First, they show that the cops-and-robber games associated with both measures are non-monotone, which draws a contrast with the case of treewidth whose associated cops-and-robber games have been shown to be monotone. Second, they show that several problems which are polynomially solvable for DAGs are still NP-complete even for graphs of constant Kelly-width and DAG-width. Yet more hardness results are shown in [17] where the Minimum Leaf Outbranching problem is shown to be NP-complete even for constant width, even though it is polynomially solvable on DAGs. Although [16,17] do not touch on the issue of parameterized hardness, as we do later on, we believe that these results fit very nicely together with the results of this paper, since they serve to underline even more the algorithmic contrast between treewidth and its directed counterparts.

A related measure is directed pathwidth. Just as pathwidth is a restriction of treewidth in the undirected case directed pathwidth is a restriction of all the previously mentioned directed measures, thus having even greater algorithmic potential. However, to the best of our knowledge no such results have been shown for directed pathwidth. In [18] it is shown that a cops-and-robber game is equivalent to directed pathwidth and that there always exists an (almost) optimal monotone strategy. It is worthy of note that, unlike the undirected case where treewidth and pathwidth are generalizations of different graph topologies (trees and paths respectively) in the directed case all the measures we have mentioned are based on the concept of DAGs as the simplest case.

Finally, it is worth noting the existence of a related digraph complexity measure which is often overlooked in this discussion: cycle rank. Cycle rank was first defined in the 1960s in [19] and it has mainly found applications in the context of regular languages (it has been shown that the star height of a regular language is connected to the cycle rank of the NFAs which accept it). As pointed out in [20] cycle rank is also relevant in our discussion here, since it can be shown [21] that the directed pathwidth of a graph is upper bounded by its cycle rank.

1.2. Our contribution

In this paper we try to address the question of whether the already proposed digraph complexity measures will be able to match the success of treewidth. Our answer is given in the form of two negative results, which show that the lack of FPT

algorithms for Directed Hamiltonian Circuit and Max Di Cut is not due to a lack of effort, but because such algorithms cannot exist (under some widely believed complexity assumptions).

Our first result concerns Directed Hamiltonian Circuit which we show to be $W[2]$-hard when the parameter is the width of the input graph for any of the mentioned widths. Under the assumption that $W[2] \neq$ FPT this implies that no *FPT* algorithm is possible. Therefore, under this standard complexity assumption, our result implies that no significant improvement is possible for the $O(n^{f(k)})$ algorithms of [13,15].

Our hardness result has immediate implications for the Minimum Leaf Outbranching problem, as Directed Hamiltonian Circuit (more precisely Directed Hamiltonian Path) is a special case of Minimum Leaf Outbranching, specifically the case where we are looking for an outbranching with exactly one leaf. Thus, it follows that Minimum Leaf Outbranching parameterized by the directed treewidth of the input graph is $W[2]$-hard even when we are looking for a solution with only a constant number of leaves. A modification of our reduction allows us to also prove a $W[2]$-hardness result for the symmetric parameterization of Minimum Leaf Outbranching, namely when the problem is parameterized by the number of leaves and the input graphs are restricted to constant directed treewidth. Informally, we can see Minimum Leaf Outbranching as a problem with two natural potential parameters, the number of leaves k and the width of the input graph w, and our results can be interpreted as meaning that both k and w must appear in the exponent of n in the running time of an algorithm for this problem. This observation fits nicely with the result of [17] where it is shown that Minimum Leaf Outbranching is in P when both k and w are constants.

Our second result concerns Max Di Cut, for which we show APX-hardness even when we restrict the problem to DAGs and all edges have uniform weights. This is a result that is interesting in its own right, and it is rather surprising that it was not known until now, as Max Di Cut is a widely studied problem. It is also very relevant in our case for two reasons: First, DAGs have the lowest possible width for all the widths we have mentioned, therefore our proof implies that none of them can help with Max Di Cut. Second, using (undirected) treewidth leads to efficient FPT algorithms for both Max Cut and Max Di Cut. Thus, this result helps draw further contrast between the performance of treewidth and its directed variants.

Although our results are negative, they succeed in illuminating some fundamental weaknesses in the already proposed digraph measures, and thus they show the way to a possible future digraph measure that might be able to overcome them. Therefore, we believe that they serve as a starting point in a renewed search for a successful digraph complexity measure that might yet manage to at least partially match treewidth's success. We refer the reader also to the recent paper [22] which discusses this very subject.

The rest of this paper is structured as follows: In Section 2 we give some necessary definitions and preliminary notions. In Section 3 we demonstrate the hardness result for Directed Hamiltonian Circuit. In Section 4 we prove the hardness of Max Di Cut. Finally, in Section 5 we conclude with some discussion and directions to further research.

2. Definitions and preliminaries

First, let us give the definitions of the two problems that will be our focus.

Definition 1. The Directed Hamiltonian Circuit problem is that of deciding whether there exists a permutation $(v_1, v_2, \ldots, v_n)$ of the vertices of an input digraph $G(V, E)$ s.t. $\forall i \in \{1, \ldots, n-1\}$ $(v_i, v_{i+1}) \in E$ and $(v_n, v_1) \in E$.

Definition 2. The Minimum Leaf Outbranching problem is the following: given a directed graph $G(V, E)$, find an outbranching (a spanning rooted directed tree) such that the number of leaves of the tree is minimized [17].

Definition 3. The Max Di Cut problem is the following: given a digraph $G(V, E)$ and a weight function on the edges $w : E \to \mathbb{N}$, find a partition of V into two sets V_0 and V_1 so that the weight of the edge set $C = \{(u, v) \mid u \in V_0, v \in V_1\}$ is maximized. That is, the objective is to maximize $\sum_{e \in C} w(e)$.

Max Di Cut was shown *APX*-hard in [23]. In Section 4 we show APX-hardness for the problem's restriction to DAGs. Then we show that APX-hardness also holds for the cardinality version of the problem restricted to DAGs.

We should also give the definitions of the two problems that will be the starting points of our reductions.

Definition 4. Dominating Set is the problem of finding a minimum cardinality subset of vertices D of an undirected graph $G(V, E)$ s.t. any vertex in $V \setminus D$ has a neighbor in D.

When a vertex $u \in D$ is a neighbor of a vertex v, we will say that u dominates v. We will also follow the convention of saying that any vertex in D dominates itself. We will make use of the well-known result that Dominating Set is $W[2]$-complete when the parameter k is the size of the dominating set we are looking for [10].

Definition 5. NAE3SAT is the problem of finding a truth assignment which, for every clause of an input 3CNF formula, assigns the value true to at least one literal, and the value false to at least one literal.

We follow the convention of saying that a clause is satisfied in the NAESAT sense, or simply satisfied, when a truth assignment assigns different truth values to two of its literals. We will mainly be concerned with the maximization version of NAE3SAT where the objective is to find a truth assignment that satisfies as many clauses as possible. This variant was shown to be APX-hard in [23].

We have already mentioned that directed pathwidth can be defined in terms of a cops-and-robber game. The game's definition is the following:

Definition 6. The k-cop invisible eager robber game is the game where k cops attempt to catch an invisible robber hiding in a vertex of a digraph G. The cops are stationed on vertices of G and a cop can move by removing himself from the graph and then "landing" on any other vertex. The robber can move at any time and he is allowed to follow any directed path of G, under the condition that he does not enter vertices occupied by stationary cops.

We say that k cops have a monotone strategy to win this game when they have a strategy such that the robber can never visit a vertex previously occupied by a cop. In [18] it was shown that k cops have a monotone strategy on a graph G iff the graph has directed pathwidth k.

Kelly-width, DAG-width and directed treewidth have also been shown to be connected to similar games, restricted to monotone strategies. In fact, DAG-width is equivalent to the above game but with the robber being visible, while Kelly-width is equivalent to the above game but with the robber only being allowed to move when a cop enters his vertex. Using the approximate connection between directed treewidth and a similar game it was shown in [15] that the directed treewidth of a graph is upper bounded by its Kelly-width multiplied by a constant.

It is not hard to infer from these results that, since the robber is stronger in the game related to directed pathwidth, a graph G will have higher pathwidth than any of the other widths. Since we are interested in proving hardness results, it will therefore suffice to show that a problem is hard for graphs of small directed pathwidth and hardness for the other widths will directly follow.

In addition to the above widths we may also wish to consider cycle rank. Cycle rank can be defined inductively as follows: if $G(V, E)$ is acyclic then $cr(G) = 0$, if G is strongly connected then $cr(G) = 1+\min_{v\in V} cr(G-v)$ and finally if G is not strongly connected then $cr(G)$ is equal to the maximum cycle rank of any of G's strongly connected components. As mentioned, it has been shown in [21] that in any digraph G the cycle rank is lower bounded by the directed pathwidth (more precisely, this relation holds up to an additive constant), therefore showing a hardness result for bounded cycle rank immediately implies hardness for all the widths we have mentioned. For the sake of completeness here is another short proof of the relation between cycle rank and directed pathwidth.

Lemma 1. *For any digraph G, $dpw(G) \leq cr(G) + 1$, where $dpw(G)$ denotes the directed pathwidth of G and $cr(G)$ the cycle rank of G.*

Proof. By induction on $cr(G)$. If $cr(G) = 0$ then G is acyclic and $dpw(G) \leq 1$. Suppose that the relation is true for all graphs of cycle rank up to k. Consider a graph G with $cr(G) = k + 1$. If it is strongly connected then there exists a vertex v such that $cr(G - v) = k$. From the inductive hypothesis this implies $dpw(G - v) \leq k + 1$, which means that $k + 1$ cops have a wining monotone strategy for $G - v$. Then $k + 2$ cops have a winning strategy for G: just keep the extra cop in v at all times. If G is not strongly connected there must exist an ordering of its strongly connected components, so that edges with endpoints in different components are always directed towards components later in the ordering. Applying the same argument to each component in this order we obtain a monotone winning strategy for the cops, because at any time the robber can either remain in the component he currently is (where the cops have a strategy) or move to a component later in the ordering (which means he can never come back). □

3. Directed Hamiltonian Circuit

In this section we focus on the Directed Hamiltonian Circuit problem, a problem which can be solved using directed treewidth in $O(n^{f(k)})$ time [13]. Of course this algorithm also applies to DAG-width, Kelly-width and directed pathwidth, as they are restrictions of directed treewidth. In addition, another $O(n^{f(k)})$ algorithm for this problem tailored for Kelly-width is given in [15]. Thus, a significant gap exists between the performance of treewidth, which is fixed-parameter polynomial on the corresponding undirected problem and the performance of its directed variants. We show that this is a gap that cannot be bridged unless $W[2] = FPT$, by demonstrating that Directed Hamiltonian Circuit is $W[2]$-hard when the parameter is any of these widths.

The hardness proof for Directed Hamiltonian Circuit will be a parameterized reduction from the naturally parameterized version of Dominating Set.

Theorem 1. *The parameterized versions of* Directed Hamiltonian Circuit, *where the parameter is the directed treewidth, Kelly-width, DAG-width, directed pathwidth or cycle rank of the input graph, are $W[2]$-hard.*

Proof. We will show a parameterized reduction from the naturally parameterized version of Dominating Set, where the parameter k is the size of the set by constructing a digraph whose cycle rank is bounded by a function of k s.t. the digraph will be Hamiltonian iff the original graph had a dominating set of size k.

Suppose we are given a graph $G(V, E)$ with $V = \{1, 2, \ldots, n\}$ and need to decide whether G has a dominating set of size k. Note that we assume that V is ordered in some way. The ordering may be arbitrary, as long as we fix it in the beginning.

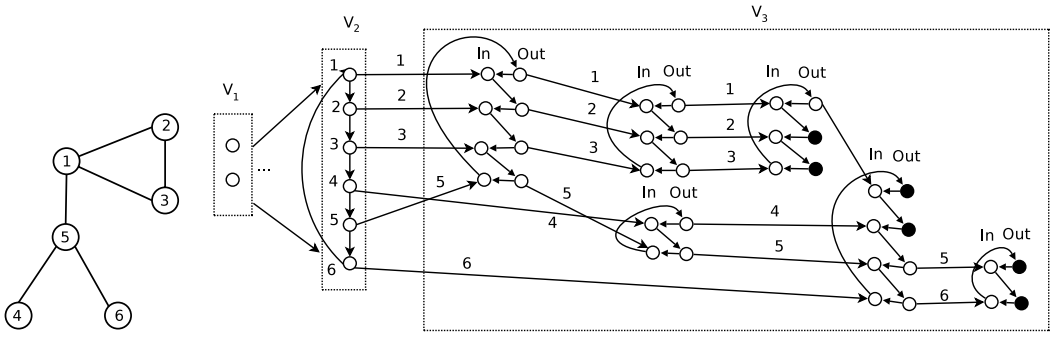

Fig. 1. An example of our construction. The graph on the left is the original undirected graph and we are looking for a dominating set of size 2. The three parts of the produced graph are outlined. To simplify the figure some edges are not shown: the dark vertices of V_3 are the vertices which are connected to all the vertices of V_1. The gadgets C_1, C_2 and C_3 are on top in V_3, while C_4, C_5 and C_6 are below.

Our digraph G' has vertex set $V' = V_1 \cup V_2 \cup V_3$ where

1. $V_1 = \{u_1, u_2, \ldots, u_k\}$.
2. $V_2 = \{v_1, v_2, \ldots, v_n\}$.
3. $V_3 = \{g_{(w,j,p)} \mid w \in \{1, \ldots, n\}, j \in \mathcal{N}[w], p \in \{In, Out\}\}$. Here $\mathcal{N}[w]$ denotes the neighborhood of vertex w in G including w itself (i.e. all the vertices that w dominates). $\mathcal{N}[w]$ is an ordered set according to the above mentioned ordering. It is also supplied with the constants f_w and l_w which denote the first and the last elements in $\mathcal{N}[w]$ respectively and the operation $s_w(j)$ which outputs the element that comes after j in $\mathcal{N}[w]$ according to the ordering.

$E(G')$ consists of the following sets of directed edges

1. $E_1 = \{(u_i, v_j) \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, n\}\}$.
2. $E_2 = \{(v_i, v_{i+1}) \mid i \in \{1, \ldots, n\}\}$ (where we consider $n + 1$ to be the same as 1).
3. $E_3 = \{(g_{(w,j,In)}, g_{(w,s_w(j),Out)}) \mid w \in \{1, \ldots, n\}, j \in \mathcal{N}[w]\}$, where $s_w(l_w) = f_w$.
4. $E_4 = \{(g_{(w,j,Out)}, g_{(w,j,In)}) \mid w \in \{1, \ldots, n\}, j \in \mathcal{N}[w]\}$.
5. Finally, E_5 contains the following edges: For any vertex w of the original graph G, the edge $(v_w, g_{(f_w,w,In)})$ and the edges $(g_{(j,w,Out)}, g_{(s_w(j),w,In)})$ for all $j \in \mathcal{N}[w]$ are included in E_5. Finally, the edges $(g_{(l_w,w,Out)}, u_i)$ for all $i \in \{1, \ldots, k\}$ are also included in E_5. We will call the subgraph induced by the group of vertices $g_{(w,j,p)}$ for a specific w, the gadget C_w.

Let us now discuss the basic idea behind this construction, before we get into more details. Our digraph G' consists of three parts: a constraint part V_1, a choice part V_2 and a satisfaction part V_3. V_1 functions as a constraint part because it only has k vertices and the only edges going into V_2 originate here, thus forcing us to enter the choice part exactly k times. A Hamiltonian tour will leave V_2 k times. The vertices from which it leaves V_2 must be (as we will prove) a dominating set of G, and that is why V_2 is the choice part. Finally, V_3 is arranged in such a way that it can only be traversed in a Hamiltonian way if the choice made in V_2 is indeed a dominating set.

It is clear that every gadget C_w is a directed cycle of $2 \cdot |\mathcal{N}[w]|$ vertices. Furthermore, the gadget C_i is connected to the gadget C_j iff there exists a vertex w in the original graph such that $i, j \in \mathcal{N}[w]$ and $j = s_w(i)$. Also notice that all edges between gadgets connect vertices having the same second coordinate and any vertex v_w of V_2 is only connected with vertices of the gadgets having w as the second coordinate.

Fig. 1 gives an example of our construction and makes it clear how the edges of E_5 are placed. For example, consider the edges we place for vertex 5 of the original graph. $\mathcal{N}[5] = \{1, 4, 5, 6\}$ in the original graph. So we must have a directed edge from v_5 to C_1, from C_1 to C_4, from C_4 to C_5, from C_5 to C_6 and C_6 back to both vertices of V_1. In order to do so we connect the vertices of each gadget that correspond to 5. Such a vertex exists in every gadget C_1, C_4, C_5, C_6 according to the construction of V_3.

The crucial part of this reduction is the way the gadget C_w works. Notice that the gadget's vertices induce a directed cycle. Also, the only way to enter this cycle is through an *In* vertex, and the only way to leave is through an *Out* vertex. Suppose that a Hamiltonian tour enters a gadget C_w m times and that $X \subseteq \mathcal{N}[w]$ is the index set of the *In* vertices which were used. Then it must also be the index set of the *Out* vertices used. To see that, suppose that $X = \{j_1, j_2, \ldots, j_m\}$ in increasing order. When entering from $g_{(w,j_1,In)}$ the tour has no choice but to proceed to $g_{(w,s_w(j_1),Out)}$. Then if $s_w(j_1) \neq j_2$ the tour must move to $g_{(w,s_w(j_1),In)}$, because if it were to exit, it would be impossible to visit $g_{(w,s_w(j_1),In)}$ in the future. Using this argument again can exclude the possibility of this part of the tour exiting through any vertex other than $g_{(w,j_2,Out)}$. Similarly, the path that starts at $g_{(w,j_2,In)}$ will exit at $g_{(w,j_3,Out)}$ and so on, with $g_{(w,j_m,In)}$ exiting through $g_{(w,j_1,Out)}$. This procedure covers all the vertices of the gadget, therefore we proved that, for any set of entry vertices X, the gadget can be traversed in a way that does not exclude the existence of a Hamiltonian tour of the whole graph iff X corresponds also to the exit vertices used.

Suppose that G does not have a dominating set of size k, but that a Hamiltonian tour of G' exists. As noticed, a Hamiltonian tour will traverse V_2 in total k times. Let D be the set of choices made by the tour in V_2, i.e. the set of vertices through which the tour exits V_2. The selection of the set D in G leaves some vertex not dominated, say vertex w. Consider the gadget C_w.

Since the tour is Hamiltonian, the gadget C_w should be traversed. Suppose that the Hamiltonian tour enters C_w through vertex $g_{(w,q,In)}$. That means that q belongs in $\mathcal{N}[w]$. Combining the previously established properties of the gadgets, the Hamiltonian tour enters and exits every gadget using only vertices having second coordinates from the set D. From this we conclude that q belongs to D. Thus, we have a contradiction since D dominates w.

It remains to prove the converse, namely that a dominating set of size k implies a Hamiltonian tour. Let $D = \{d_1, d_2, \ldots, d_k\}$ be a dominating set. Informally, these will be exactly the vertices through which our tour will exit V_2. Also, because of the construction of the C_w gadgets, if such a gadget through a set of In vertices it is possible to traverse it in a Hamiltonian way exiting exactly from the same set of corresponding Out vertices. Keeping that into account we will have to show that all C_w gadgets are entered at least once, but that follows from the fact that D is a dominating set.

Let us first describe the tour outside the gadgets. Starting at u_1, move to v_{d_k+1} (once again, v_{n+1} is the same as v_1) and then follow the edges in V_2 until v_{d_1} is reached. Then we exit V_2 towards the gadgets. When we reach an *Out* gadget vertex that points to V_1 we move to u_2. From there we move to v_{d_1+1}, then to v_{d_2} and so on. This procedure makes sure that, even though we enter V_2 only k times, all of its n vertices are covered.

Let us now describe the traversal of the gadgets, starting from gadget 1. First note that $D \cap \mathcal{N}[1] \neq \emptyset$ because D is a dominating set. It is not hard to see that our tour will enter gadget C_1 through vertices $g_{(1,d_j,In)}$ for all j such that $d_j \in D \cap \mathcal{N}[1]$, since $f_j = 1$ for all these j and we leave V_2 only from exit points corresponding to D. Once inside the gadget at the vertex $g_{(1,d_j,In)}$ our tour follows the unique path to $g_{(1,d_l,Out)}$, where d_l is the next element of $D \cap \mathcal{N}[1]$ according to the ordering (or the first element of $D \cap \mathcal{N}[1]$ if d_j is the last). Note that if $|D \cap \mathcal{N}[1]| = 1$ then $d_l = d_j$. Thus, all vertices of gadget C_1 are visited exactly once and the gadget is exited through vertices corresponding to $D \cap \mathcal{N}[1]$.

We will now inductively prove the same for all gadgets. Suppose that for all gadgets up to gadget C_i we have shown that all their vertices are visited exactly once and the gadgets are entered and exited through vertices corresponding to $D \cap \mathcal{N}[i]$. Let us now consider gadget C_{i+1}. Once again $D \cap \mathcal{N}[i+1] \neq \emptyset$ because D is a dominating set. The gadget C_{i+1} is entered only through vertices $g_{(i+1,d_j,In)}$ such that $d_j \in D \cap \mathcal{N}[i+1]$ because the only edges going into gadget C_{i+1} originate in V_2 or one of the previous gadgets for which we have assumed that they are exited through vertices corresponding to D. Once inside gadget C_{i+1} we follow a similar tour as in gadget 1; starting from vertex $g_{(i+1,d_j,In)}$ we follow the unique path to $g_{(i+1,d_l,Out)}$ and leave the gadget, where d_l is the element of $D \cap \mathcal{N}[i+1]$ which comes after d_j (or the first element of $D \cap \mathcal{N}[i+1]$ if d_j is the last). With the same reasoning as previously, all vertices of gadget C_{i+1} are visited exactly once and the gadget is exited only through vertices corresponding to $D \cap \mathcal{N}[i+1]$. This completes the proof that a Hamiltonian tour can be constructed.

Finally, what is left is to argue is that G' has low width.

First, notice that $cr(G') \leq |V_1| + cr(G' - V_1) = k + cr(G' - V_1)$. But $G' - V_1$ is not strongly connected and all its strongly connected components are directed cycles (V_2 and the gadgets C_i). Therefore, $cr(G' - V_1) \leq 1$. □

We modify the above reduction in order to prove the following theorem

Theorem 2. *The parameterized version of* Minimum Leaf Outbranching *where the parameter is the number of the leaves of the outbranching is W*[2]*-hard even when restricted to graphs with constant cycle rank.*

Proof. We reduce the Dominating Set problem to the Minimum Leaf Outbranching problem. We modify the construction of the graph G' of Theorem 1, adding a vertex r with arcs pointing to the k vertices of set V_1 and deleting those edges of E_5 that connect the gadgets with V_1. We name the new graph G''.

Vertex r will definitely serve as the root of the outbranching since it is a source. We prove that G has a dominating set of size k iff G'' has an outbranching with k leaves.

Suppose that there is a dominating set of size k in G. From Theorem 1 we have that G' has a Hamiltonian cycle which uses k edges of E_5 that connect V_3 with V_1. Those edges are missing from G''. Therefore there are k disjoint paths from V_1 to V_3 that cover all the vertices of G'. Thus there is an outbranching with root r with k leaves.

Furthermore suppose that G'' has an outbranching T with at most k leaves. Notice that, since r is its root and there are no arcs from V_2 or V_3 to V_1, all the k arcs from r to V_1 are contained in T. Thus there are exactly k disjoint paths in T, thus exactly k leaves. Notice that if the k leaves are vertices of V_3 that connect to V_1 in G' then from T we can construct a Hamiltonian circuit in G', which can help find a dominating set of size k in G (by Theorem 1). Name these vertices of V_3 that connect to V_1 in G' black vertices. We prove that from any outbranching T with k leaves we can construct an outbranching T' with k black leaves.

First of all we can assume that T has no leaves in V_1 or V_2. If there was a leaf u_i in V_1 and the arc (v_{j-1}, v_j) is part of T then we could add the arc (u_i, v_j) and remove the arc (v_{j-1}, v_j) from T so u_i would not be a leaf anymore. If there was a leaf v_j in V_2, following a similar procedure as above we could add the arc $(v_j, g_{(f_j,j,In)})$ and remove the arc $(g_{(f_j,j,Out)}, g_{(f_j,j,In)})$, so v_j would not be a leaf anymore. Furthermore notice that there is no way that a vertex $g_{(i,w,In)}$ could be a leaf since vertex $g_{(s_w(i),w,Out)}$ could not be reached. So wlog we can assume that all the leaves of T are out vertices of the gadgets.

We show by induction on the gadgets that we can eliminate all non-black leaves from T. For every gadget i starting from gadget 1 up to $n-1$ we eliminate all the black leaves from gadget i. Suppose that there is a leaf $g_{(i,w,Out)}$ in T which is not a black vertex. Then the arc $(g_{(i,w,Out)}, g_{(s_w(i),w,In)})$ is not in T. However, vertex $g_{(s_w(i),w,In)}$ is in T, thus the arc $(g_{(s_w(i),w,Out)}, g_{(s_w(i),w,In)})$ should be in T. By removing $(g_{(s_w(i),w,Out)}, g_{(s_w(i),w,In)})$ and then adding $(g_{(i,w,Out)}, g_{(s_w(i),w,In)})$ we assure that $g_{(i,w,Out)}$ is not a leaf anymore while making sure that this procedure does not create non-black leaves in gadgets $1 \ldots i-1$. We repeat the procedure until no non-black leaf exists in gadget i. Then we continue with gadget $i+1$. Finally the last gadget n cannot have a non-black leaf since all its Out vertices are black.

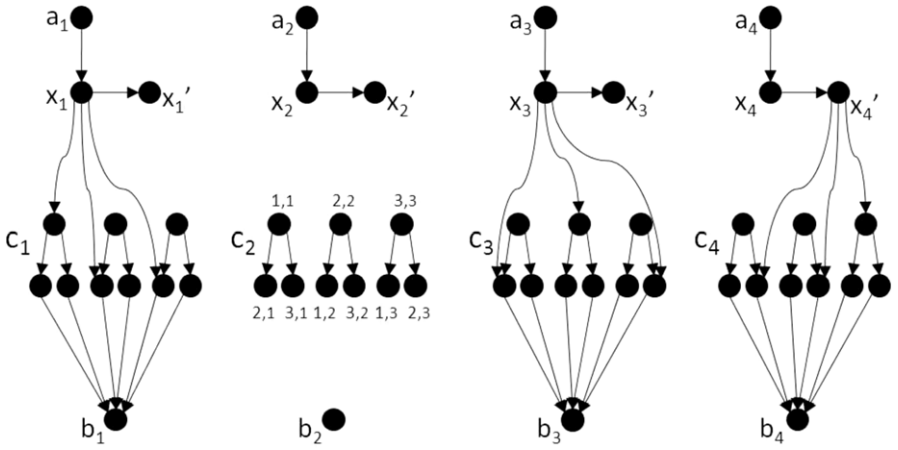

Fig. 2. The above figure presents an example of the construction for the formula $\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee x_4)$. From ϕ we construct $\phi' = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_3 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$. In order for the figure to be understandable we excluded most of the edges of E_5 together with some edges of E_3.

Furthermore, G'' has constant cycle rank since it is not strongly connected and all the strongly connected components are cycles which have constant cycle rank. □

4. Maximum Directed Cut

Let us now focus on a problem of much different nature: Max Di Cut. Even though, as we saw in Section 3, no digraph complexity measure manages to provide an FPT algorithm for Directed Hamiltonian Circuit, they do succeed in providing algorithms with polynomial running times, when the width k is fixed. For Max Di Cut the situation is much worse, as we will show that the problem is NP-hard even for $k = 1$. This creates an even larger gap with the FPT performance of treewidth than we had in the case of Directed Hamiltonian Circuit.

We will prove that Max Di Cut is both NP- and APX-hard, even when restricted to DAGs by showing a reduction from the maximization version of NAE3SAT.

Theorem 3. Max Di Cut *is NP-hard and APX-hard, even when restricted to DAGs.*

Proof. We give a gap-preserving reduction from NAE3SAT to Max Di Cut.

Given a NAE3SAT formula ϕ with m clauses and n variables we construct a new NAE3SAT formula ϕ' with $2m$ clauses and n variables and show that ϕ is satisfiable iff ϕ' is satisfiable (satisfaction is in the NAESAT sense). Then from ϕ' we construct a (weighted) DAG G and show that ϕ' is satisfiable iff G has a directed cut of size $46m$. Without loss of generality, we may assume that every clause of ϕ has exactly three literals (otherwise we may repeat one).

The new formula ϕ' is constructed by taking ϕ and adding to it, for every clause the same clause with all literals complemented. If an assignment satisfies t clauses of the original formula, it must satisfy exactly $2t$ of the $2m$ clauses of ϕ'. Note that, if we denote by f_i the number of appearances of the variable x_i in ϕ, then the same variable will appear $2f_i$ times in ϕ': f_i times as x_i and f_i times as $\neg x_i$. In other words, the positive and negative appearances of each variable in ϕ' are balanced. We will make use of this fact several times. Furthermore, since every clause of ϕ has exactly three literals, we have that $\sum_i f_i = 3m$

Let us now construct the DAG $G(V, E)$. V consists of four disjoint sets of vertices A, X, C, B. $A = \{a_1, \ldots, a_n\}$ will be a set of source vertices. $B = \{b_1, \ldots, b_{2m}\}$ will be a set of sink vertices. $X = \{x_1, x'_1, x_2, x'_2, \ldots, x_n, x'_n\}$ will be the set of vertices corresponding to literals of ϕ' while $C = \{c_{i,j,k} \mid i \in \{1, 2, \ldots, 2m\},\ j, k \in \{1, 2, 3\}\}$ will correspond to the clauses of ϕ'.

E consists of the following sets of weighted edges:

1. The set $E_1 = \{(a_i, x_i) \mid i \in \{1, \ldots, n\}\}$. Each of these edges has weight $6f_i$, where f_i is the total number of appearances of the variable x_i in ϕ.
2. The set $E_2 = \{(x_i, x'_i) \mid i \in \{1, \ldots, n\}\}$. Each of these edges also has weight $6f_i$.
3. The set $E_3 = \{(c_{i,j,k}, b_i) \mid i \in \{1, \ldots, 2m\},\ j, k \in \{1, 2, 3\}, j \neq k\}$. These have weight 1.
4. The set $E_4 = \{(c_{i,k,k}, c_{i,j,k}) \mid i \in \{1, \ldots, 2m\},\ j, k \in \{1, 2, 3\}, j \neq k\}$. These also have weight 1.
5. Finally, E_5 consists of edges that connect vertices of the set X to the corresponding vertices of C. That is, we add the edges $\{(x_l, c_{i,j,k}),\ k \in \{1, 2, 3\}\}$ when the literal x_l appears in the j-th position of the i-th clause of ϕ', and the edges $(x'_l, c_{i,j,k})$ when the literal $\neg x_l$ appears in that position. These edges have weight 2.

An illustrative example of the construction is presented in Fig. 2. Vertex x'_4 is connected to $c_{4,3,1}$, $c_{4,3,2}$ and $c_{4,3,3}$ since $\neg x_4$ appears in the third position of the fourth clause. The intuition behind our construction is that the placement of the

vertices of C on either side of the cut will correspond to the truth assignments for the literals. The edges inside C take care of the satisfaction. For each clause we construct three triplets of vertices. Each triplet corresponds to a different arrangement of the literals in the specific clause, where in each arrangement a different literal of the clause is placed on top and this retains the symmetry in the clauses. Specifically, the vertex $c_{i,j,k}$ corresponds to the j-th literal of the i-th clause of ϕ' in the arrangement where the k-th literal is placed on top. In a satisfied clause one literal is false and one true and there is always an arrangement which places the true literal on top and the false one on bottom, thus contributing to the cut. However for non-satisfied clauses none of the arrangements contribute to the cut.

Suppose we are given a truth assignment that satisfies (in the NAESAT sense) t of the m clauses of ϕ. It must satisfy $2t$ of the $2m$ clauses of ϕ'. Let us partition V into V_0 and V_1. Place all vertices of A into V_0 and all vertices of B into V_1. Place the vertices of X that correspond to true literals in V_1 and the rest in V_0. Place the vertices of C that correspond to true literals in V_0 and the rest in V_1.

Let us calculate the weight of this cut. If a variable x_i is assigned the value 1 in the assignment, the edge (a_i, x_i) contributes $6f_i$ to the cut. If it is assigned 0, then x'_i is in V_1, therefore the edge (x_i, x'_i) contributes $6f_i$ to the cut. Thus, the total contribution of all edges in $E_1 \cup E_2$ is $6\sum_i f_i = 18m$. Because the appearances of each variable in ϕ' are balanced, there are as many literals that took the value true as there are literals that took the value false, in any assignment. Therefore, exactly half the edges of E_3 contribute to the cut. The number of edges in E_3 is $12m$ so, a weight of $6m$ is contributed to the cut. It is not hard to see that, for a satisfied clause C_i, the edges of E_4 incident on vertices that correspond to this clause contribute exactly 2 to the cut. On the other hand, the edges of E_4 incident on vertices of C that correspond to a clause that is not satisfied will contribute 0 to the cut, since all these vertices correspond to literals with the same truth value and are therefore on the same side of the partition. Thus, we get a total of $4t$ contributed to the cut, since $2t$ clauses are satisfied. Finally, once again because of the balancing of ϕ', exactly half of the edges of E_5 contribute to the cut: those incident on vertices of X that we placed in V_0, i.e. vertices that correspond to false literals. For each such element of X we have in total $3f_i$ edges. Since the weight of each such edge is 2, this adds up to a total contribution of $6\sum_i f_i = 18m$.

Thus, the total size of the cut is $18m + 6m + 18m + 4t = 42m + 4t$, which is equal to $46m$ when the truth assignment satisfies every clause of ϕ.

Now for the other direction, suppose we are given a partition of V into V_0 and V_1. We will show that we can transform such a cut into a cut of the previous form, thus obtaining a truth assignment. First, observe that for any optimal cut $A \subseteq V_0$ and $B \subseteq V_1$, because it is always optimal to place a source in V_0 and a sink in V_1. Now, suppose that in the cut we are given, for some $i, x_i, x'_i \in V_0$. Then place x'_i in V_1 and this will not make the cut smaller because now the edge (x_i, x'_i) contributes to the cut and its weight is exactly as much as the weight of all other edges incident on x'_i. Also, if $x_i, x'_i \in V_1$ place x_i in V_0. This cannot make the cut smaller, since the only edge lost is (a_i, x_i) and its weight is the same as that of (x_i, x'_i) which now enters the cut. Therefore, we have now made sure that for all i, x_i and x'_i are on different sides of the partition, without decreasing the size of our cut.

Consider now a vertex $c_{i,j,j}$. We know that there exists an edge $(x_i, c_{i,j,j})$ (or an edge $(x'_i, c_{i,j,j})$) of weight 2, which is as much as the weight of all other edges incident on $c_{i,j,j}$. Therefore, if x_i (resp. x'_i) is in V_0, then we can place $c_{i,j,j}$ in V_1 without decreasing the size of the cut. Otherwise, we can place $c_{i,j,j}$ in V_0, because the edge of weight 2 cannot be included in the cut by changing the side of $c_{i,j,j}$ only, and therefore placing it in V_0 is not worse because this way we may also include some of the other edges in the cut. This establishes that every vertex $c_{i,j,j}$ is on a different side of the partition from its predecessor in X.

Finally, consider a vertex $c_{i,j,k},\ j \neq k$. If its predecessor in X is in V_0 we can place it in V_1 without decreasing the size of the cut, because then the edge of weight 2 is included. Otherwise we can place it in V_0, and this will include the edge $(c_{i,j,k}, b_i)$ in the cut. This does not decrease the size of the cut, since the edge of weight 2 was not included anyway, therefore we might at most lose the other edge incident on this vertex, which also has weight 1. This establishes that each of the remaining vertices of C is also on a different side of the partition from its predecessor in X.

Now, observe that starting with any given cut, we have transformed it into a cut of a special form, without decreasing its size. From this cut we can construct a truth assignment: set to true the literals corresponding to vertices in X that we placed in V_1. This is a valid assignment, since exactly one of x_i, x'_i is in V_1. Also, if we repeat the process of the first direction of this reduction starting from this assignment we will get the same cut. Therefore, we have shown that there is a truth assignment that satisfies t of the m clauses of ϕ iff there is a cut in the DAG G of size at least $42m + 4t$. Thus,

$$OPT_{NAESAT}(\phi) = m \Rightarrow OPT_{MDC}(G) = 46m$$

$$OPT_{NAESAT}(\phi) \leq (1-\epsilon)m \Rightarrow OPT_{MDC}(G) \leq \left(1 - \frac{2\epsilon}{23}\right) 46m. \quad \square$$

It is not hard to extend the results of the previous theorem to the cardinality version of Max Di Cut, that is, the version where all edges have the same weight.

Theorem 4. *Cardinality* Max Di Cut *is NP- and APX-hard, even when restricted to DAGs.*

Proof. First, observe that all the edge weights used in the proof of Theorem 3 are polynomially (in fact linearly) bounded by the size of the original NAE3SAT formula. Thus, if we extend the problem's definition to include multigraphs, we can replace every edge of weight w by w parallel edges of weight 1. It is not hard to see that this does not affect the rest of the proof.

Now, let us show how to eliminate parallel edges. For each edge (u, v) introduce a directed path of length 3 u, w_1, w_2, v where w_1 and w_2 are new vertices. Observe that, if u is assigned 0 and v is assigned 1, then it is possible to include 2 of the 3 edges of the path in the cut, by assigning 0 to w_2 and 1 to w_1. However, any other assignment to u and v ensures that at most 1 of the three edges can be included in the cut, and in fact this is always possible by assigning 0 to w_1 and 1 to w_2. Thus, it is not hard to see that the reduction's arguments can now be applied with little modification. □

Corollary 1. Max Di Cut *is NP-hard and APX-hard even when restricted to graphs of bounded directed treewidth, DAG-width, Kelly-width, directed pathwidth or cycle rank.*

Proof. The proof is immediate, because DAGs have width at most 1 under the definitions of all these widths. □

5. Conclusions and further work

In this paper we have presented two hardness results affecting all known generalizations of treewidth to digraphs as well as directed pathwidth and cycle rank. It may be worthwhile at this point to discuss why such results hold for the directed cousins of treewidth, when in the undirected case there has been such a huge success.

First, the hardness result for Max Di Cut, gives us one indication why such hardness results hold. The reason is simply that for some problems DAGs are not really an "easy" topology, as trees are in the undirected case. The fact that DAGs are not as easy as trees has been more or less known for years and in this sense it is quite surprising that essentially all of the research on directed variants of treewidth has so far taken the approach of generalizing DAGs. A further clue is given in this direction by the fact that DAGs are the base case for both directed pathwidth and the three treewidth variants we considered. One would probably expect pathwidth and treewidth to be based on different graph topologies.

On the other hand, simply discarding DAGs as a starting point does not seem like a good solution as directed treewidth variants have had some success with path-based problems, such as Directed Hamiltonian Circuit. For such problems, DAGs usually are indeed the trivial case and it makes sense to design a width as a generalization of DAGs. However, we showed that none of the currently known widths (including directed pathwidth) is restrictive enough to provide an FPT algorithm for Directed Hamiltonian Circuit.

Therefore, we believe that our results may suggest that in the directed case things may be more complicated and possibly no "right" complexity measure exists. On one hand, it would probably make sense to explore the possibility of a width (not based on DAGs) that can solve Max Di Cut and similar problems, while still being more general than undirected treewidth. And on the other hand, a more realistic goal might be to attempt to refine the definition of some of the already known widths (which are based on DAGs) in order to make it restrictive enough to solve Directed Hamiltonian Circuit and related path problems in FPT time.

Acknowledgement

We would like to thank Hermann Gruber for stimulating discussion on the subject and for pointing out the existence of cycle rank to us.

References

[1] Neil Robertson, Paul D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, J. Algorithms 7 (3) (1986) 309–322.
[2] Paul D. Seymour, Robin Thomas, Graph searching and a min–max theorem for tree-width, J. Combin. Theory, Ser. B 58 (1) (1993) 22–33.
[3] Nick D. Dendris, Lefteris M. Kirousis, Dimitrios M. Thilikos, Fugitive-search games on graphs and related parameters, Theoret. Comput. Sci. 172 (1–2) (1997) 233–254.
[4] Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, Detlef Seese, An algebraic theory of graph reduction, J. ACM 40 (5) (1993) 1134–1164.
[5] Hans L. Bodlaender, Treewidth: Structure and algorithms, in: Giuseppe Prencipe, Shmuel Zaks (Eds.), SIROCCO, in: Lecture Notes in Computer Science, vol. 4474, Springer, 2007, pp. 11–25.
[6] Hans L. Bodlaender, Treewidth: Characterizations, applications, and computations, in: Fedor V. Fomin (Ed.), WG, in: Lecture Notes in Computer Science, vol. 4271, Springer, 2006, pp. 1–14.
[7] Hans L. Bodlaender, Treewidth: Algorithmic techniques and results, in: Igor Prívara, Peter Ruzicka (Eds.), MFCS, in: Lecture Notes in Computer Science, vol. 1295, Springer, 1997, pp. 19–36.
[8] Dániel Marx, Can you beat treewidth? in: FOCS, IEEE Computer Society, 2007, pp. 169–179.
[9] Bruno Courcelle, The monadic second-order logic of graphs. I. Recognizable sets of finite graphs, Inf. Comput. 85 (1) (1990) 12–75.
[10] Rod G. Downey, M.R. Fellows, Parameterized Complexity, Springer, 1999.
[11] Rolf Niedermeier, Oxford Lecture Series in Mathematics and Its Applications, Oxford University Press, USA, 2006.
[12] J. Flum, M. Grohe, Parameterized Complexity Theory, 1 edition, in: Texts in Theoretical Computer Science. An EATCS Series, Springer, 2006.
[13] Thor Johnson, Neil Robertson, Paul D. Seymour, Robin Thomas, Directed tree-width, J. Combin. Theory, Ser. B 82 (1) (2001) 138–154.
[14] Jan Obdrzálek, Dag-width: connectivity measure for directed graphs, in: SODA, ACM Press, 2006, pp. 814–821.
[15] Paul Hunter, Stephan Kreutzer, Digraph measures: Kelly decompositions, games, and orderings, in: Nikhil Bansal, Kirk Pruhs, Clifford Stein (Eds.), SODA, SIAM, 2007, pp. 637–644.
[16] Stephan Kreutzer, Sebastian Ordyniak, Digraph decompositions and monotonicity in digraph searching, in: Hajo Broersma, Thomas Erlebach, Tom Friedetzky, Daniël Paulusma (Eds.), WG, in: Lecture Notes in Computer Science, vol. 5344, 2008, pp. 336–347.
[17] P. Dankelmann, G. Gutin, E.J. Kim, On complexity of minimum leaf out-branching problem, Discrete Appl. Math. 157 (13) (2009) 3000–3004.
[18] János Barát, Directed path-width and monotonicity in digraph searching, Graphs Combin. 22 (2) (2006) 161–172.
[19] L.C. Eggan, Transition graphs and the star-height of regular events, Michigan Math. J 10 (4) (1963) 385–397.

[20] Hermann Gruber, Markus Holzer, Finite automata, digraph connectivity, and regular expression size, in: Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, Igor Walukiewicz (Eds.), ICALP (2), in: Lecture Notes in Computer Science, vol. 5126, Springer, 2008, pp. 39–50.
[21] Hermann Gruber, Markus Holzer, Finite automata, digraph connectivity, and regular expression size, Technical Report, 2007. http://drehscheibe.in.tum.de/forschung/pub/reports/2007/TUM-I0725.pdf.gz.
[22] Robert Ganian, Petr Hlinený, Joachim Kneis, Alexander Langer, Jan Obdrzálek, Peter Rossmanith, On digraph width measures in parameterized algorithmics, in: Jianer Chen, Fedor V. Fomin (Eds.), IWPEC, in: Lecture Notes in Computer Science, vol. 5917, Springer, 2009, pp. 185–197.
[23] Christos H. Papadimitriou, Mihalis Yannakakis, Optimization, approximation, and complexity classes, J. Comput. System Sci. 43 (3) (1991) 425–440.